Using Automatic Differentiation in Sensitivity Analysis of Nuclear Simulation Models

Mihai Alexe[1], Oleg Roderick, Mihai Anitescu, Jean Utke[2], Thomas Fanning, and Paul Hovland

Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439
mihai@vt.edu, roderick@mcs.anl.gov, anitescu@mcs.anl.gov, utke@mcs.anl.gov, fanning@anl.gov, hovland@mcs.anl.gov

INTRODUCTION

Sensitivity analysis is an important tool in the study of nuclear systems. In our recent work, we introduced a hybrid method that combines sampling techniques with first-order sensitivity analysis to approximate the effects of uncertainty in parameters of a nuclear reactor simulation model. For elementary examples, the approach offers a substantial advantage (in precision, computational efficiency, or both) over classical methods of uncertainty quantification [1].

First-order sensitivity analysis is predicated on obtaining derivative information from nuclear engineering models. For software suites such as SAS4A/SASSYS-1, manual development of derivative information is considered to be a difficult task [2]. Divided-difference approaches are inefficient for high-dimensional uncertainty sets. Automatic differentiation (AD) approaches [3,4] offer a conceptually promising alternative for obtaining derivatives.

Most of the available AD tools are developed for research purposes; their potential benefit for realistic applications has not been fully examined. We refer to [5] for an example of applying AD to Fortran 90 code, but we note that many models of interest are more complex and developed according to older (Fortran 77) standards.

We investigate the following question: Are AD tools now at a stage where they can provide derivative information for realistic nuclear engineering codes at a minor development cost?

DESCRIPTION OF THE WORK

AD is a technique used to evaluate derivatives of functions defined by computer programs based on the fact that any program can be viewed as a finite sequence of elementary operations, the derivatives of which are known.

A program P implementing the function $\Im = f(\alpha)$, can be parsed into a sequence of elementary steps:

$$P: \Im = f_k(f_{k-1}(...f_1(\alpha)...)) \qquad (1)$$

The task of AD is to assemble a new program P' to compute the derivative.

$$P': \left(\nabla_\alpha \Im\right)_i = \frac{\partial f_k}{\partial f_{k-1}} \cdot \frac{\partial f_{k-1}}{\partial f_{k-2}} \cdot ... \cdot \frac{\partial f_1}{\partial \alpha_i}, \qquad (2)$$
$$i = 1,2,...$$

The *reverse* (adjoint) mode of AD reverses the control flow of the program to compute the derivative.

$$\nabla_\alpha \Im = \left(\frac{\partial f_1}{\partial \alpha}\right)^T \cdot \left(\frac{\partial f_2}{\partial f_1}\right)^T \cdot ... \cdot \left(\frac{\partial f_k}{\partial f_{k-1}}\right)^T \qquad (3)$$

The advantage of the adjoint mode over the *direct* mode (2) is computational efficiency: it allows computation of all components of the gradient in a single run of P', as opposed to multiple runs required by (2). There is a theoretical limit, *independent of the uncertainty dimension*, on the additional computational cost of running the AD-augmented code in reverse mode [4].

In an ideal situation, preparing the code for processing by AD tools is limited to identifying inputs and outputs of interest. In practice, application of AD requires additional effort.

Applying AD to Codes with Major Legacy Components

The following (Fortran 77) programming features potentially make the application of AD difficult:

- Machine-dependent code sections need to be removed.
- EQUIVALENCE constructs need to be replaced by simple array allocations and element-wise assignments.
- COMMON blocks with inconsistent sizes between subroutines need to be renamed; in addition, initialization with strong typing needs to be enforced through use of additional variables in each subroutine.
- Subroutines with variable number of parameters need to be split into separate subroutines for each type of call.

1 Virginia Tech
2 ANL; Computation Inst., Univ. of Chicago.

- Direct memory copy operations need to be rewritten as explicit operations.
- IMPLICIT definitions occasionally need to be replaced by subroutine definitions.

We note that the problematic features encountered so far are not related to the mathematical structure of the model. In the possible case of inherently non smooth operations, a smoothing interpolation can be differentiated instead, which also does not require modifications of the mathematical structure. We estimate that a competent programmer would need approximately one week to prepare a code of comparable complexity for automatic differentiation. Since the results of sensitivity analysis have many applications, we find the development cost acceptable. We hope to provide a detailed description of performance and development in an extended version of the paper.

Applied Example

We applied automatic differentiation tools (in forward mode) to a nuclear reactor simulation code known as MATWS. The model combines the point-kinetics module from the SAS4A/SASSYS-1 code with a simplified representation of the reactor heat removal systems. The code consists of over 10,000 lines of Fortran 77. MATWS was previously used in combination with the statistical simulation tool GoldSim to model accident scenarios [2].

The chosen outputs of interest were the transient coolant, cladding, structure, and fuel temperatures at the end of a simulation for an unprotected loss of flow. The chosen inputs were the radial core expansion, control rod driveline expansion, fuel axial expansion, and Doppler reactivity feedback coefficients. The code was passed through different AD tools: ADIFOR, OpenAD/F, and TAMC in direct mode and TAPENADE in reverse mode [6]. The tools use somewhat different algorithms for optimal evaluation of the sequence of numerical operations required by (2) or (3). For verification, we also computed the derivatives using finite-difference approximation, FD.

Table 1. Comparison of different estimates for the derivatives of the fuel and coolant temperatures with respect to the radial core expansion coefficient.

AD Tool	Fuel Temperature Derivative, K	Coolant Temperature Derivative, K
ADIFOR	18312.5474227	17468.4511373
OpenAD/F	18312.5474227	17468.4511372
TAMC	18312.5474248	17468.4511392
TAPENADE	18312.5474227	17468.4511372
FD	18312.5269537	17468.4315994

RESULTS

Table 1 presents a portion of derivative verification tests. All the results and FD approximation agree within 0.01% or better (and all AD results agree almost perfectly with each other). We conclude that AD results are valid.

We view our work as a strong argument for applying AD to nuclear engineering codes with legacy components. This opens the way to improving a wide range of existing tools of simulation and analysis in nuclear reactor applications. An ideal subject of AD-based sensitivity analysis is a code that is portable, language-standard compliant, and reversible (effectively, does not contain features that make adjoint differentiation inefficient). Not all the models can be held to this standard. Future work on using AD for sensitivity analysis will likely be a co-design process, with improvements made to AD tools and the newly developed models simultaneously.

ACKNOWLEDGMENTS

This work was supported by the U.S. Dept. of Enegry under Contract No. DE-AC02-06CH11357.

REFERENCES

1. O. RODERICK, M. ANITESCU, and P. FISCHER, “Polynomial regression approaches using derivative information for uncertainty quantification,” Nuclear Science and Engineering, **164** (2), pp 122-139 (2010).
2. E. MORRIS and W. NUTT, “Uncertainty analysis for unprotected loss-of-heat-sink, loss-of-flow, and transient-overower events in sodium-cooled fast reactors,” in Proc. International Conference on Fast Reactors and Related Fuel Cycle (2009).
3. J. UTKE, U. NAUMANN, M. FAGAN, N. TALLENT, M. STROUT, P. HEIMBACH, C. HILL, and C. WUNSCH, “OpenAD/F: A modular open-source tools for automatic differentiation of Fortran codes,” ACM Trans. Math. Softw., **34**, pp. 1-36 (2008).
4. A. GRIEWANK, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Frontiers in Applied Mathematics, **19** (2000).
5. B. REARDEN and J. HORWEDEL, “Automatic differentiation with code coupling and applications to scale modules,” in Proc. Joint International Topical Meeting on Mathematics and Computation and Supercomputing in Nuclear Applications (2007).
6. M. ALEXE, O. RODERICK, J. UTKE, M. ANITESCU, P. HOVLAND and T. FANNING, “Automatic differentiation of codes in nuclear engineering applications,” Tech. Rep. ANL/MCS-310 Argonne (2009).